AMENDMENTS TO THE CLAIMS

Claims 3, 11 and 13 have been canceled without prejudice.

This listing of Claims shall replace all prior versions, and listings, of claims in the application:

LISTING OF CLAIMS:

1. (previously presented) A boot method for an In-Circuit Emulation system having a microcontroller operating in lock-step synchronization with a virtual microcontroller, comprising:

in the microcontroller, executing a set of boot code;

in the virtual microcontroller, executing a set of timing code to enable the lock-step synchronization, wherein the timing code is timed to take the same number of clock cycles as the microcontroller uses to execute the boot code, and wherein the boot code is stored within the microcontroller and at least one portion of the boot code is inaccessible to the virtual microcontroller; and

simultaneously halting both the microcontroller and the virtual microcontroller.

2. (original) The method according to claim 1, further comprising copying register contents from the microcontroller to corresponding registers in the virtual microcontroller.

3. (canceled)

4. (original) The method according to claim 1, wherein after the executing of the boot code, the microcontroller branches to assembly instruction line 0; and wherein after executing the timing code, the virtual microcontroller branches to assembly instruction line 0.

5. (original) The method according to claim 1, wherein prior to the executing of the boot code, and prior to executing the timing code, a break is set at assembly instruction line 0.

6. (previously presented) The method according to claim 1, wherein the boot code comprises protected initialization code that is not accessible to the In-Circuit Emulation system.

7. (original) The method according to claim 1, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at assembly instruction line 0; and

wherein after the executing of the boot code the microcontroller branches to assembly instruction line 0; and

wherein after executing the timing code, the virtual microcontroller branches to assembly instruction line 0.

8. (original) The method according to claim 1, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at assembly instruction line 0;

wherein after the executing of the boot code, the microcontroller branches to assembly instruction line 0; and wherein after executing the timing code, the virtual microcontroller branches to assembly instruction line 0;

copying register contents from the microcontroller to corresponding registers in the virtual microcontroller;

copying memory contents from the microcontroller to corresponding memory in the virtual microcontroller;

wherein after the executing of the boot code, the microcontroller branches to assembly instruction line 0; and

wherein after executing the timing code, the virtual microcontroller branches to assembly instruction line zero.

9. (original) The method according to claim 8, further comprising removing the break at assembly line zero after copying the register contents and copying the memory contents.

10. (previously presented) A boot method for an In-Circuit Emulation system having a microcontroller operating in lock-step synchronization with a virtual microcontroller, comprising:

resetting the microcontroller and the virtual microcontroller to a halt state; setting a break at assembly instruction line zero;

in the microcontroller, executing a set of boot code;

in the virtual microcontroller, executing a set of timing code to enable the lock-step synchronization, wherein the timing code is timed to take the same number of clock cycles as the microcontroller uses to execute the boot code, and wherein the boot code is stored within the microcontroller and at least one portion of the boot code is inaccessible to the virtual microcontroller;

simultaneously halting both the microcontroller and the virtual microcontroller by branching to assembly instruction line zero;

copying register contents from the microcontroller to corresponding registers in the virtual microcontroller;

copying memory contents from the microcontroller to corresponding memory in the virtual microcontroller; and

removing the break at assembly line zero after copying the register contents and copying the memory contents.

11. (cancelled)

12. (previously presented) A boot method for an In-Circuit Emulation system having a device under test operating in lock-step synchronization with a virtual processor, comprising:

in the device under test, executing a set of boot code;

in the virtual processor, executing a set of timing code to enable the lock-step synchronization, wherein the timing code is timed to take the same number of clock cycles as the device under test uses to execute the boot code, wherein

the boot code is stored within the device under test and at least one portion of the boot code is inaccessible to the virtual processor; and

simultaneously halting both the device under test and the virtual processor.

13. (canceled)

14. (previously presented) The method according to claim 12, further comprising copying memory contents from memory coupled to the device under test to corresponding memory coupled to the virtual processor.

15. (original) The method according to claim 12, wherein after the executing of the boot code, the device under test branches to assembly instruction line 0; and wherein after executing the timing code, the virtual processor branches to assembly instruction line 0.

16. (original) The method according to claim 12, wherein prior to the executing of the boot code, and prior to executing the timing code, a break is set at assembly instruction line 0.

17. (previously presented) The method according to claim 12, wherein the boot code comprises protected initialization code that is not accessible to the In-Circuit Emulation system.

18. (original) The method according to claim 12, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at assembly instruction line 0; and

wherein after the executing of the boot code, the device under test branches to assembly instruction line 0; and

wherein after executing the timing code, the virtual processor branches to assembly instruction line 0.

19. (previously presented) The method according to claim 12, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at assembly instruction line 0;

wherein after the executing of the boot code, the device under test branches to assembly instruction line 0; and wherein after executing the timing code, the virtual processor branches to assembly instruction line 0;

copying register contents from the device under test to corresponding registers in the virtual processor;

copying memory contents from the device under test to corresponding memory in the virtual processor;

wherein after the executing of the boot code, the device under test branches to assembly instruction line 0; and

wherein after executing the timing code, the virtual processor branches to assembly instruction line zero.

20. (original) The method according to claim 19, further comprising removing the break at assembly line zero after copying the register contents and copying the memory contents.

21. (original) The method according to claim 12, wherein the virtual processor is implemented in a field programmable gate array.